Modeling and Characterizing Shared and Local Memories of the Ampere GPUs

Hamdy Abdelkhalik[1], Yehia Arafa[1,3], Nandakishore Santhi[2], Nirmal Prajapati[2], and Abdel-Hameed A. Badawy[1,2]
[1] Klipsch School of ECE, New Mexico State University, Las Cruces, NM 80003, USA
{enghamdy, yarafa, badawy}@nmsu.edu
[2] Los Alamos National Laboratory, Los Alamos, NM 87545, USA
{nsanthi, prajapati}@lanl.gov
[3] Qualcomm Inc, USA

ABSTRACT

The rapid evolution of GPU architectures necessitates advanced modeling techniques for optimization and understanding their intricate functionalities. The Performance Prediction Toolkit for GPUs (PPT-GPU) is an innovative modeling tool developed to deliver detailed modeling of GPU architectures. It allows researchers and developers to understand, analyze, and predict the behavior of different GPU components under various computational loads. PPT-GPU is invaluable in enabling detailed insights into GPU architectures, such as memory performance metrics, total active cycles, and utilization of GPU resources.

This paper extends PPT-GPU to model the NVIDIA Ampere architecture. Specifically, we focus on modeling the shared memory and the register spilling represented by the local memory operations. We have used several performance metrics to validate our work against Nvidia Nsight. Additionally, our refined version of PPT-GPU offers detailed metrics, capturing active cycles of functional units, an essential aspect in analyzing application performance constraints. The enhanced PPT-GPU has an average prediction error of 14.6% for total active cycles and 13% and 13.3% for the L1 and L2 hit rates, respectively.

CCS CONCEPTS

• **Hardware** → **Hardware accelerators**; • **General and reference** → *Performance*;

KEYWORDS

GPU Modeling, Ampere Architecture, PPT-GPU, GPU shared memory, GPU local memory

ACM Reference Format:
Hamdy Abdelkhalik[1], Yehia Arafa[1,3], Nandakishore Santhi[2], Nirmal Prajapati[2], and Abdel-Hameed A. Badawy[1,2]. 2023. Modeling and Characterizing Shared and Local Memories of the Ampere GPUs. In *The International Symposium on Memory Systems (MEMSYS '23), October 2–5, 2023, Alexandria, VA, USA*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The advent of advanced Graphics Processing Units (GPUs) like Nvidia Ampere [14] architecture has brought forth a new era of computational power and efficiency. To fully leverage these capabilities and optimize GPU-based applications, there is a need for precise modeling and understanding of such complex architectures. In this study, we have significantly enhanced a tool known as PPT-GPU [4, 6] to facilitate comprehensive modeling of the Ampere architecture, specifically focusing on shared and local memories.

The A100 GPU [13], as a representation of the cutting-edge Ampere architecture, embodies significant innovations in its on-chip memory systems and computational units. The new design of the shared memory (Fig. 1) presented in Ampere architecture achieves high optimization via asynchronous copy operations, which eliminate idleness by facilitating simultaneous data transmission and computations and can bypass the L1 cache to save bandwidth and time. The A100 GPU also provides local memory as an additional resource for storing thread-specific data when register capacities are exhausted, thereby balancing data processing speed and storage requirements.

Before our enhancements, PPT-GPU was accurately modeling Volta and Turing architectures [4]. However, accurate modeling of the Ampere architecture, with its distinctive shared memory structure, demanded extensions to PPT-GPU.

Initially, the instructions latencies for Ampere were estimated [2]. This was a fundamental step to set the stage for subsequent modeling efforts. With these latencies and other specific configurations as input, we model the new design of the shared and local memories using our extended PPT-GPU implementation.

We compared our results against Nvidia Nsight [15] to uphold the integrity and corroborate the efficacy of PPT-GPU. Additionally, to corroborate the dependability of our tool, we ran a diverse set of benchmarks, extending from linear algebra (e.g., 2MM, BICG, etc.) to machine learning applications (e.g., Deepbench).

2 RELATED WORK

GPGPU-Sim [7], an early tool, provided cycle-level simulations for GPUs from Fermi to Pascal, but its slow speed limited its use in large-scale projects. Barra [8], meanwhile, offered functional CUDA program simulations but needed more depth in capturing architectural details. Wong *et al.* presented the Multi2Sim tool [18], focusing on AMD's architecture. This tool offered a combined CPU and GPU simulation. Kerr et al. developed a model primarily for predicting

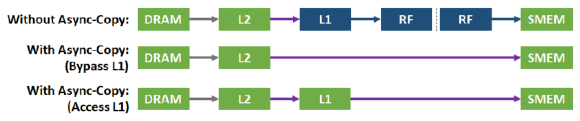

Figure 1: Shared memory in Ampere [13].

CUDA data transfer times without covering broad architectural behaviors [9]. The introduction of the Accel-Sim framework [10] marked a significant step forward, catering to contemporary GPU architecture research, but its coverage halted at the Volta architectures. Demonstrated on the A100 GPU, this study significantly pushes the boundaries beyond previous works, uniquely covering the innovative design of the Ampere architecture, and stands as the first to provide a comprehensive model for the said architecture.

3 PPT-GPU

The Performance Prediction Toolkit for GPUs (PPT-GPU) is an integral part of the Performance Prediction Toolkit (PPT), an open-source project [5] developed at Los Alamos National Laboratory. PPT-GPU is a specialized tool designed to facilitate in-depth modeling of complex GPU architectures and predicts their performance. It relies on the SASS [12] instruction traces collected using NVBit [19] to capture the dynamic behavior of the application accurately.

We undertake an expansive enhancement of PPT-GPU in response to the need for a more intricate understanding of the Ampere architectures. In the extended PPT-GPU, we make significant adjustments to cater to the new shared memory design (see Figure 1). The updated model now recognizes that asynchronous copy operations in shared memory follow two distinct hardware paths, resulting in different SASS instructions. One path allows the data to flow from the global memory through the L1 and L2 caches before going to the shared memory (*LDGSTS.E*). The other path, identified by the "Bypass" operator, facilitates direct data movement from the global memory to the shared memory (*LDGSTS.BYPASS.E*).

To model these operations, we collect two different memory traces for L1 and L2 caches (in the same run) and recognize that certain operations use L2 while bypassing L1. These traces are then integrated with the traces of other standard memory operations. Subsequently, we utilize these traces to calculate the hit rates for both L1 and L2 caches by leveraging existing reuse distance profiling methods in PPT-GPU. We also estimate the latency of the direct path from the global to shared memory via L2 to be about *170 cycles*, compared to *300 cycles* for the full path. However, this latency is hidden as memory operations exchange with the computational operations.

Local memory plays a significant role in influencing application performance, making its accurate modeling critical for performance prediction and optimization efforts [11, 16, 20]. Local memory uses the same data path as normal memory operations. It takes part of the global memory and can be cached in the L1 and L2 caches. Local memory has a thread-private scope similar to the register file [11]. We gather address traces on a per-block granularity for the normal global memory operations. Whereas for the local memory, we collect the traces per warp. Moreover, our analysis found that the local memory addresses differ from those of other memory operations. For instance, the standard global operations consistently possess 12 hexadecimal digits, and the local memory addresses are represented with only six digits (e.g., *0X7F088EA22000 vs. 0XFFFC44*). We couple this analysis with the traces collected from load and store operations of local memory reuse distance profiling to estimate the hit rates for both operations [3]. Furthermore, we accurately compute the number of requests and transactions throughout the kernel execution.

Table 1: Local memory metrics error percentages for three applications from the Cutlass benchmark [17].

%Error	FP64_Gemm	TF32_Symm	Gemm_Softmax
Local LD hit rate	0	4	8
Local ST hit rate	0	6	15
Local trans	0	0	3
Local Requests	0	1	5

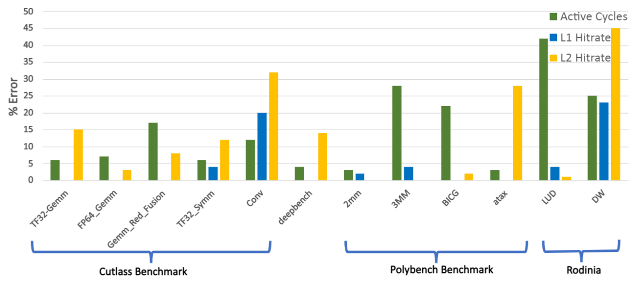

Figure 2: Prediction results.

The enhanced PPT-GPU now integrates a wider range of additional metrics. Among these are local hit rates, transactions, requests, and the measurement of active cycles for Tensor Cores and various other compute units. Such details aid in providing an in-depth understanding of the inner workings of the GPU. This ultimately assists in pinpointing performance bottlenecks and paves the way for more efficient optimization of code specific to the Ampere architecture.

4 EXPERIMENTAL EVALUATION

We ran experiments using Cuda 11.7.0 on an A100 GPU. We experimented with a wide range of benchmarks, from simple kernels like matrix multiplication to large kernels from Deepbench [1], a machine learning application leveraging tensor cores and shared memory. We also cross-verified our results against Nvidia Nsight.

Figure 2 highlights three pivotal performance metrics: total active cycles and hit rates for L1 and L2 caches. The first six applications incorporate the updated operations and structure of the shared memory. For instance, *FP64_Gemm* instructions bypass the L1 cache, while instructions in *TF32_Gemm* access it. The rest use a combination of these operations.

Finally, we have three applications that use register spilling, which results in local memory operations (*FP64_Gemm*, *TF32_Symm*, and *Gemm_Softmax*). These observations highlight that these applications effectively leverage the two principal components we integrated into PPT-GPU.

Figure 2 shows the percentage error in prediction for active cycles, L1 hit rate, and L2 hit rate. The average error in the prediction of hit rates for L1 and L2 caches are 13% and 13.3%, respectively. Furthermore, in our evaluation of PPT-GPU, the error in total active cycles across all applications ranges from a minimum of 3% to a maximum of 42%, with an average of 14.6% error. Table 1 summarizes the local memory results. The accuracy of all metrics for the *FP64_Gemm* is 100%. For the other two applications, the prediction error in hit rates of LD and ST operations ranges from 4% to 15%.

5 CONCLUSIONS

In this paper, we modeled the Nvidia Ampere shared and local memories in-depth and substantially enhanced the PPT-GPU to depict its local memory and the novel design of its shared memory. We used a broad set of benchmarks to validate PPT-GPU and compared the results against Nvidia Nsight. The active cycles, a primary performance metric, have an average error rate of 14.6%, while the L1 and L2 cache memories hit rates stand at 13% and 13.3%, respectively.

REFERENCES

[1] DeepBench. https://svail.github.io/DeepBench/.

[2] Hamdy Abdelkhalik, Yehia Arafa, Nandakishore Santhi, and Abdel-Hameed A. Badawy. 2022. Demystifying the Nvidia Ampere Architecture through Microbenchmarking and Instruction-level Analysis. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–8. https://doi.org/10.1109/HPEC55821.2022.9926299

[3] Yehia Arafa, Abdel-Hameed Badawy, Gopinath Chennupati, Atanu Barai, Nandakishore Santhi, and Stephan Eidenbenz. 2020. Fast, Accurate, and Scalable Memory Modeling of GPGPUs Using Reuse Profiles. In *Proceedings of the 34th ACM International Conference on Supercomputing (ICS '20)*. Association for Computing Machinery, New York, NY, USA, Article 31, 12 pages. https://doi.org/10.1145/3392717.3392761

[4] Yehia Arafa, Abdel-Hameed Badawy, Ammar ElWazir, Atanu Barai, Ali Eker, Gopinath Chennupati, Nandakishore Santhi, and Stephan Eidenbenz. 2021. Hybrid, scalable, trace-driven performance modeling of GPGPUs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.

[5] Yehia Arafa, Abdel-Hameed A. Badawy, Gopinath Chennupati, Nandakishore Santhi, and Stephan Eidenbenz. PPT-GPU Tool. https://github.com/lanl/PPT

[6] Yehia Arafa, Abdel-Hameed A. Badawy, Gopinath Chennupati, Nandakishore Santhi, and Stephan Eidenbenz. 2019. *IEEE Computer Architecture Letters* 18, 1 (2019), 55–58. https://doi.org/10.1109/LCA.2019.2904497

[7] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt. 2009. Analyzing CUDA workloads using a detailed GPU simulator. In *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. 163–174. https://doi.org/10.1109/ISPASS.2009.4919648

[8] Caroline Collange, Marc Daumas, David Defour, and David Parello. 2010. Barra: A parallel functional simulator for gpgpu. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 351–360.

[9] Andrew Kerr, Gregory Diamos, and Sudhakar Yalamanchili. 2010. Modeling GPU-CPU workloads and systems. In *Proceedings of the 3rd workshop on general-purpose computation on graphics processing units*. 31–42.

[10] Mahmoud Khairy, Zhesheng Shen, Tor M Aamodt, and Timothy G Rogers. 2020. Accel-Sim: An extensible simulation framework for validated GPU modeling. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 473–486.

[11] Ang Li, Shuaiwen Leon Song, Akash Kumar, Eddy Z. Zhang, Daniel Chavarría-Miranda, and Henk Corporaal. 2016. Critical points based register-concurrency autotuning for GPUs. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1273–1278.

[12] NVIDIA. CUDA Binary Utilities. https://docs.nvidia.com/cuda/cuda-binary-utilities/

[13] NVIDIA. NVIDIA A100 whitepaper. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

[14] NVIDIA. NVIDIA Ampere Architecture In-Depth | NVIDIA Developer Blog. https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

[15] NVIDIA. NVIDIA Nsight Systems. https://developer.nvidia.com/nsight-systems

[16] Putt Sakdhnagool, Amit Sabne, and Rudolf Eigenmann. 2019. *arXiv preprint arXiv:1907.02894* (2019).

[17] Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. 2023. CUTLASS. https://github.com/NVIDIA/cutlass.

[18] Rafael Ubal, Byunghyun Jang, Perhaad Mistry, Dana Schaa, and David Kaeli. 2012. Multi2Sim: A simulation framework for CPU-GPU computing. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*. 335–344.

[19] Oreste Villa, Mark Stephenson, David Nellans, and Stephen W Keckler. 2019. Nvbit: A dynamic binary instrumentation framework for nvidia gpus. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 372–383.

[20] Xiaolong Xie, Yun Liang, Xiuhong Li, Yudong Wu, Guangyu Sun, Tao Wang, and Dongrui Fan. 2015. Enabling Coordinated Register Allocation and Thread-Level Parallelism Optimization for GPUs. In *Proceedings of the 48th International Symposium on Microarchitecture (MICRO-48)*. Association for Computing Machinery, New York, NY, USA, 395–406. https://doi.org/10.1145/2830772.2830813